VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

PROHLÍŽEČ FOTOGRAFIÍ S 3D AKCELERACÍ

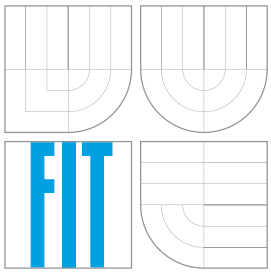
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VERONIKA GAJOVÁ

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

PROHLÍŽEČ FOTOGRAFIÍ S 3D AKCELERACÍ

3D ACCELERATED PHOTO VIEWER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VERONIKA GAJOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR POSPÍCHAL

BRNO 2010

Abstrakt

Tato práce se zabývá problémem návrhu a implementace jednoduchého multiplatformního prohlížeče fotografií, využívajícího grafickou knihovnu OpenGL. Výstupem je interaktivní prezentace fotografií ve virtuálním prostoru za použití 3D akcelerace grafickou kartou počítače. Zobrazení fotografií se dosáhne nanášením příslušných obrazových dat jako textur na polygony o odpovídajících poměrech stran. Protože je vyžadována podpora více grafických formátů, pro načítání obrazových dat je využita knihovna DevIL. Plynulosti chodu aplikace je dosaženo použitím vláken. Zobrazovací vlákno plynule vykresluje scénu, zatímco druhé vlákno načítá požadovaná obrazová data. Pro korektní práci s vlákny a přístup ke sdíleným proměnným jsou využity vhodné synchronizační mechanismy.

Abstract

This work deals with the problem of design and implementation of simple multiplatform picture viewer, using OpenGL graphic library. Viewing of pictures is set in ambience of 3D scene and takes advantage of 3D hardware acceleration. Displaying of pictures is implemented by application of image data as textures on polygons with specific aspect ratio. The image library DevIL is used for loading image data, because the support of wide variety of graphic file formats is required. Lag-free execution of the aplication is achieved by usage of multiple threads. The first thread displays the scene and the second one loads the required image data. Suitable synchronization mechanisms must be used for the correct work of threads and access to shared variables.

Klíčová slova

prohlížeč fotografií, grafika, 3D akcelerace, fotografie, textury, vlákna, OpenGL, GLUT, DevIL, SDL

Keywords

picture viewer, graphics, 3D acceleration, pictures, textures, threads, OpenGL, GLUT, DevIL, SDL

Citace

Veronika Gajová: Prohlížeč fotografií s 3D akcelerací, bakalářská práce, Brno, FIT VUT v Brně, 2010

Prohlížeč fotografií s 3D akcelerací

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Petra Pospíchala. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Veronika Gajová
16. května 2010

Poděkování

Děkuji Ing. Petrovi Pospíchalovi za cenné rady, trpělivost a čas. Děkuji svým rodičům za podporu při dosavadním studiu.

Obsah

Kapitola 1

Úvod

S rychlým rozvojem digitální fotografie během několika posledních let zákonitě úzce souvisí vývoj programů, které umožňují práci s obrázky. Značná část uživatelů využívá fotoaparát pro zhotovování převážně dokumentárních snímků (z akcí, dovolených) a zamýšlí provádět základní úpravy fotografií. Dále ocení zejména možnosti organizace, prohlížení a prezentace. Poutavost prostředí je zaručena postupně se rozmáhajícím 3D zpracováním grafiky, které se projevuje zejména v oblasti prohlížení a prezentace snímků, anebo také například v grafických uživatelských rozhraních operačních systémů.

V poslední době je rovněž výrazným trendem vystavování a tedy i snadnější sdílení fotografií či videí na internetu na specializovaných serverech typu Picasa, Flickr, Youtube, případně sociálních sítích. Statické stránky internetových galerií postrádají prezentační atraktivitu, a navíc zahrnují neefektivnost vlivem procházení obrázků pomocí klikání na ikony, čekání na načítání apod. Objevuje se proto software sloužící pro zobrazování internetových galerií a pohlížení fotografií umístěných na internetu. Příkladem je rozšíření pro internetové prohlížeče Cooliris anebo prohlížeč FotoViewr a jemu podobné. Tyto prohlížeče obvykle vyžadují uživatelský účet v rámci internetové galerie, jako je například Flickr. S tím souvisí i potřeba spravování, nahrávání fotografií do virtuální galerie, často striktně kapacitně omezené, anebo placené.

Další kategorií prohlížečů fotografií tvoří prohlížeče fotografií uložených na pevném disku uživatelova počítače. Pro uživatele, který nepotřebuje sdílet své fotografie, jsou tyto prohlížeče více uživatelsky přívětivé. Výrazným trendem je v dnešní době integrace dostatečného množství funkcí, aby nebylo nutné používat další programy (například pro přehrávání hudby na pozadí, základní úpravy fotografií), a přitom nároky na hardware nebyly přehnané a nebyla nepříznivě ovlivněna rychlost. Například komerční program ePic, vyvinutý pro OS Microsoft Windows, realisticky modeluje prohlížení „rozsypaných“ fotografií na povrchu scény, Visions pro Windows XP umožňuje současné prohlížení fotografií z více složek současně[1].

1.1 Předmět práce

Při vývoji prohlížečů fotografií by se, obecně řečeno, měla věnovat největší pozornost zejména uživatelské přívětivosti, přiměřené atraktivitě uživatelského rozhraní a jednoduchosti ovládání. Důležitým hlediskem pro uživatele je celková kvalita v porovnání s mírou komerčnosti produktu, případně specializace na určitý operační systém.

[1] Více informací lze nalézt na domovských stránkách zmíněných produktů ([18, 17, 19, 20])

Prohlížeč fotografií by proto měl být především nekomerční, přiměřeně jednoduchý, intuitivní z hlediska ovládání, rychlý a ideálně multiplatformní. Aplikace, která je předmětem této bakalářské práce, bude zaměřena na cílovou skupinu uživatelů, kteří nepotřebují, či nezamýšlí své fotografie dodatečně upravovat, sdílet, ani organizovat, tedy takové, kteří je chtějí pouze pohodlně a atraktivně vizuálně prezentovat.

1.2 Členění práce

V textu této práce uvádím informace týkající se počítačové grafiky a 3D akcelerace, dostupných grafických knihoven a jejich porovnání (kapitola 2.1).

Rozebrána nutná teorie z oblasti grafiky a grafické knihovny OpenGL zahrnující vykreslování scény (2.4), transformace a zásobník matic (2.5, 2.6).

Ohledně vizuální stránky Prohlížeče je poměrně důležité zmínit se o barvách (2.7).

Texturování je pro tuto práci klíčové, a je mu věnována kapitola 2.8. Jsou rozebrány důvody používáni textur, princip nanášení, filtrování, používání tzv. mipmap, nahrazování textur a v neposlední řadě speciální strukturu poskytovanou OpenGL pro práci s texturami, objektem textury (2.8.3). S texturováním, respektive načítáním texturových dat rovněž souvisí zmínka o paměti PBO (2.9).

Kapitola 2.10 se zabývá otázkou dosažením maximální rychlosti a plynulosti prohlížení a načítání fotografií, zajištěné dynamickým načítáním obrázků prostřednictvím vláken.

Kapitola Návrhu (3) nastiňuje vzhled a činnost aplikace, konkrétně rozvržení fotografií ve scéně, uspořádání na listech (3.1). O představě grafického uživatelského rozhraní se zmiňuje 3.2 a je navržen způsob načítání zdrojových obrázkových souborů (3.3). Zejména je rozvržena činnost a synchronizace zobrazovacího a načítacího vlákna (3.4).

Z implementačního hlediska je rozebrána nezbytná práce se souborovým systémem, průchod při volbě zdrojové složky a identifikace obrázků (4.1).

Kapitola 4.2 objasňuje implementaci grafického uživatelského rozhraní, samotnou interakcí s uživatelem se poté zabývá 4.3.

Řešení vykreslování scény, realizaci načítání fotografií prostřednictvím knihovny DevIL a animace vysvětluje 4.4. Práce s vlákny, synchonizace, komunikace mezi vlákny a implementace prostřednictvím knihovny SDL je popsána v 4.5.

Práce a dosažené výsledky jsou závěrečně shrnuty v kapitole 5, kde jsou rovněž rozebrány možnosti dalšího rozšíření.

Kapitola 2

Grafická knihovna OpenGL

2.1 Grafika a 3D akcelerace

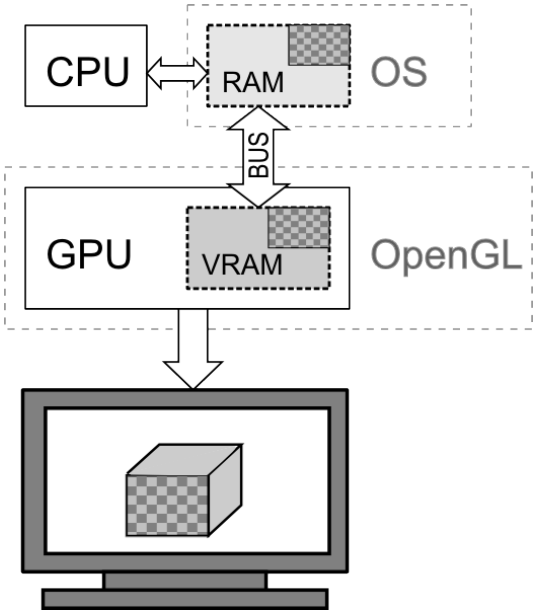
S počátkem 90. let dochází k rozšiřování 3D grafiky, zejména v počítačových hrách, a tedy potřebou vykreslování grafiky v reálném čase. Vzniká požadavek na hardwarovou 3D akceleraci, neboli přesunutí 3D výpočtů či jejich částí z procesoru počítače do grafické karty pro zpracování často výkonnějším grafickým procesorem [16], zatímco komunikaci mezi systémovou (operační) pamětí a grafickým akcelerátorem řídí CPU [10]. Následným přínosem je celkové zvýšení výkonnosti a zejména rychlosti provádění akcelerovaných 3D operací, kterými jsou například práce s geometrií 3D scény, texturováním, osvětlením a stínováním, nebo transformace polygonové sítě [8].

Tlak herního průmyslu má za následek rychlý rozvoj funkcionality grafických čipů a nárůst komplexnosti aplikací, které je využívají. Pokud jsou v rámci aplikace použity nepodporované funkce, jsou emulovány softwarově za cenu snížení výkonu.

V současnosti nejpoužívanějšímy standardy pro definici 2D a 3D grafiky je OpenGL a DirectX. Zhlediska grafiky jde konkrétně o DirectX Graphics společnosti Microsoft, určené výhradně pro operační systémy Windows. DirectX Graphics dále zahrnuje DirectDraw pro práci s rastrovou 2D grafikou a Direct3D pro práci s 3D grafikou. Od OpenGL se komponenty DirectX Graphics, vedle již zmíněné závislosti na prostředí OS Microsoft Windows, liší zejména objektovým přístupem vystavěným na COM (Component Object Model) [7]. Multiplatformnost OpenGL je jednoznačným důvodem pro volbu právě této grafické knihovny, má-li být vyvíjená aplikace přenositelná, jako je to zamýšleno i u Prohlížeče, který je předmětem této práce.

2.2 Knihovna OpenGL

Knihovna OpenGL (Open Graphics Library) firmy Silicon Graphics Inc. (SGI), je v dnešní době jedním z hlavních aplikačních programových rozhraní (API) ke grafickému hardwaru, konkrétně grafickým kartám podporujícím 3D akceleraci. Zásadní výhodou knihovny OpenGL je bezesporu to, že ji lze využívat na různých operačních systémech. Knihovna OpenGL je proto mimořádně vhodná pro vývoj multiplatformních aplikací a díky této vlastnosti bude použita při implementaci Prohlížeče. Informace v následujícím textu jsou ve značné míře čerpány z knih o OpenGL [9, 2].



Obrázek 2.1: Činnost OpenGL

Knihovna OpenGL neposkytuje systémově závislé funkce pro práci s okny, vytváření grafického uživatelského rozhraní nebo pro zpracování událostí. Pro tyto účely je nutné použít jiné prostředky, jako nadstavbu GLUT nad OpenGL, případně multimediální knihovnu SDL.

2.3 GLUT (OpenGL Utility Toolkit)

OpenGL Utility Toolkit, dále zkráceně GLUT, je systémově nezávislá programová knihovna, která definuje a implementuje aplikační rozhraní pro tvorbu oken a jednoduchého uživatelského rozhraní.

Značnou výhodou knihovny GLUT je vytváření přenositelných aplikací na úrovni zdrojového kódu, oproti přístupu, kdy je grafické uživatelské rozhraní programováno přes aplikační programové rozhraní daného operačního systému a stává se tak platformně závislým.

Další výhodou knihovny GLUT je jednoduchost programového rozhraní. To se skládá z volání funkcí vracejících pouze základní datové typy jazyka C. Nevýhodami, se kterými bude potřeba při návrhu aplikace počítat, je mírná zastaralost v oblasti podpory novějších periferních zařízení a minimalistická koncepce týkající se prvků grafického rozhraní, například nemožnost tvorby složitějších prvků jako jsou tlačítka, seznamy, editační pole apod. GLUT z tohoto hlediska podporuje pouze vytváření samostatných oken, suboken, překryvných oken a vyskakovacích menu. Mimojiné umožňuje práci s bitmapovými a vektorovými písmy [11]. .

2.4 Vykreslování scény

OpenGL neposkytuje vysokoúrovňové příkazy pro vykreslování 3D objektů, jeho model tak sestává pouze ze základních grafických primitiv. Za elementární grafické objekty se považují geometrické útvary:

bod definovaný souřadnicemi v prostoru

úsečka definovaná souřadnicemi koncových bodů

mnohoúhelník daný souřadnicemi svých vrcholů

bitmapa sestávající z jednotlivých obsažených bitů[1]

2.4.1 Stavový automat

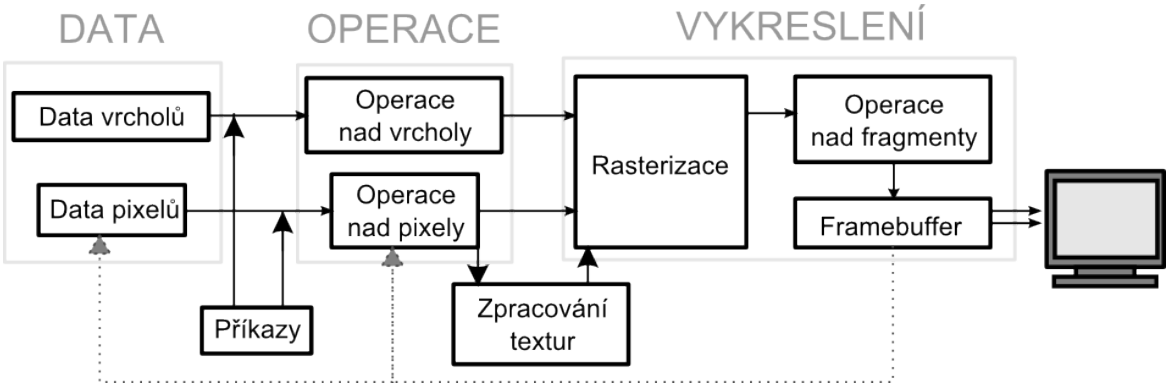
OpenGL se chová jako stavový automat, neboť během zadávání vykreslovacích příkazů zachovává stav, mód, spojený s určitými stavovými proměnnými které mohou být svázány s grafickými primitivy či celou scénou, dokud tento není explicitně změněn. V daném okamžiku se vždy nachází v určitém stavu, daném nastavením proměnných. Vykreslování je potom ovlivněno aktuálním nastavením konečného automatu. Příkladem je například situace, kdy dojde k nastavení proměnné definující vykreslovací barvu. Tato se poté používá pro všechna následná vykreslování objektů, dokud není opět explicitně změnena.

2.4.2 Průběh vykreslování

Vykreslování scény sestává z několika kroků. Nejdříve dojde ke zkonstruování tvarů objektu z grafických primitiv a je vytvořen matematický popis objektů, poté jsou umístěny do 3D prostoru a je zvolen pozorovací bod scény. Dojde k výpočtu barev objektů. Barvy jsou zadány buď přímo, nebo je ovlivňuje aktuální nastavením světelných podmínek a vlastností materiálu, či použitá textura. Nakonec proběhne proces rasterizace, neboli převedení primitiv – bodu, úsečky, polygonu nebo pixelů bitmapy na fragmenty odpovídající pixelům v bufferu.

Pro vyhodnocení viditelnosti příslušného pixelu se probíhá test hloubky. V OpenGL je implementován depth-bufferem (paměť hloubky). Zde je pro každý pixel obrazovky uložena vzdálenost od bodu pohledu k fragmentu příslušejícímu tomuto pixelu. Pokud má testovaný kandidát menší vzdálenost od pozorovatele (například náleží-li bližšímu objektu), uloží se tato hloubka na příslušnou pozici v depth-bufferu pro další porovnání. Proto dojde k vykreslování pouze viditelných objektů či viditelných částí objektů. Vykreslovacím vstupem jsou tedy grafická primitiva ve formě vykreslovacího řetězce, neboli matematického popisu objektů a barev, konečným výsledkem je poté promítnutí framebufferu na obrazovku. Informace o bodech jsou dány bitovými plochami, úseky paměti, kde pro každou plochu jeden bit náleží jednomu bodu obrazovky. Seskupené bitové plochy tvoří tzv. framebuffer (paměť snímku). Framebuffer uchovává veškeré informace pro grafický displej, týkající se řízení barvy a intenzity všech bodů na obrazovce. Posloupnost operací v rámci OpenGL nazývaná OpenGL pipeline určuje, že geometrická data (vrcholy, přímky a mnohoúhelníky) jsou zpracovávána odděleně od rastrových dat. Společně vchází do procesu rasterizace a fragmentových operací. Nakonec se výsledek zapíše do framebufferu, jak je znázorněno na obrázku 2.2 [9].

[1] Oproti předešlým primitivům je tedy bitmapa nikoli vektorovým, ale rastrovým primitivem



Obrázek 2.2: Data a operace v pipeline, upr. [9], str. 37

2.4.3 Double-buffering

Program vykresluje scénu průběžně, nikoli ve formě kompletních obrázků. Toto se může projevit například při animacích poměrně složité scény jako blikání později vykreslovaných objektů, neboť tyto jsou krátce po vykreslení opět smazány před následným překreslením scény.

Většina implementací OpenGL poskytuje hardwarový či softwarový tzv. double-buffering. Jedná se o dva kompletní buffery pro barvy, z nichž jeden je zobrazen a do druhého je zatím vykreslováno. Po vyrenderování scény se oba buffery prohodí, a následuje zobrazení nově vykresleného bufferu a vykreslování do původně zobrazeného.

2.5 Transformace v OpenGL

Umístění a vykreslení modelu ve scéně, konkrétně specifikace pixelů a způsobu jejich zobrazení, je výsledkem několika operací. Je třeba převést trojrozměrné souřadnice objektů na pozice pixelů obrazovky postupnými transformacemi. Ty jsou reprezentovanými násobením matic. Dále se provede ořezání objektů či jejich částí ležících mimo scénu a provedení zobrazovací transformace, která přiřadí transformovaným souřadnicím pixely obrazovky.

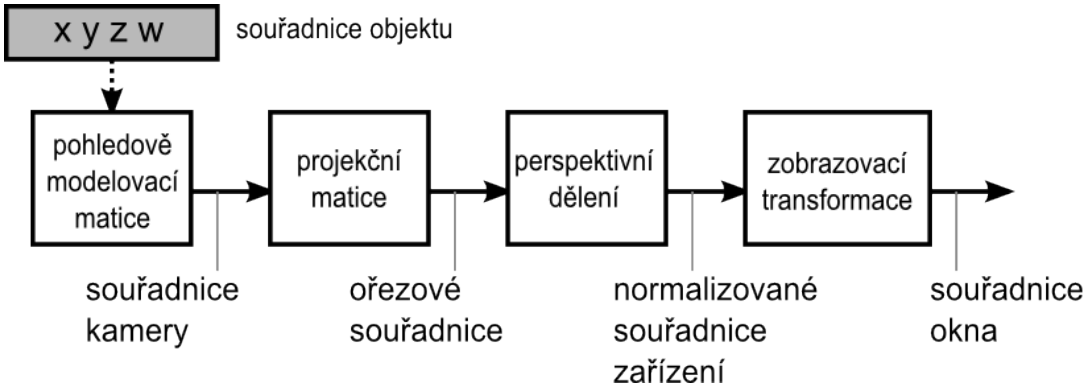
Provádí se tři základní typy transformací: pohledová, modelovací a projekční.

Transformace lze popsat příslušnými maticemi M 4×4 [2], kterou je manipulován (násoben) každý vrchol ve vykreslovacím řetězci podle transformačního vzorce

$$v' = Mv \tag{2.1}$$

Pohledová transformace (viewing transformation) provede nastavení kamery, konkrétně její pozice a směru pohledu, či orientaci souřadného systému. Po nastavení kamery následuje postavení scény (modelovací transformace) pro určení pozice a orientace objektů ve scéně, například rotace, posunutí, změna měřítka.

[2] Deklarace matice v jazyce C jako $m[4][4]$ není vhodná, neboť standardní transformační maticí OpenGL je $m[i][j]$, kde i je sloupec a j řádek. Standardní konvence v C chápe i jako řádek a j jako sloupec. Potom je vhodné definovat matice 4×4 jako $m[16]$, anebo použít analogické procedury OpenGL pro nastavování a násobení matic, které pracují s transponovanými maticemi



Obrázek 2.3: Jednotlivé kroky transformace vrcholů; up. orig.[9], str. 113

Modelovací transformace (modeling transformation) ovlivňují umístění a orientaci modelu. Základními funkcemi OpenGL pro modelovací transformace jsou funkce pro transformaci objektu (eventuelně lokálního souřadného systému souřadnic, spjatého s objektem) – posun, rotaci, změnu měřítka a zrcadlení (v případě záporné hodnoty změny měřítka). Příkazy OpenGL pro tyto transformace vytvoří příslušnou matici transformace a aplikaci této matice vynásobením s aktuální maticí.

Projekční transformace (projection transformation) udává, jak se budou modely promítat na obrazovku, přičemž lze použít dva typy projekce, ortografickou a perspektivní.

Ortografická projekce mapuje objekty na obrazovku bez úpravy jejich relativní velikosti. Projekce perspektivní odpovídá vnímání v běžném životě, kdy dochází k deformacím, zmenšování objektu se vzrůstající vzdáleností od kamery. Obě projekce se liší geometrií objemu pohledu a ořezávacích rovin. Tyto jevy lze pozorovat a porovnat na ilustraci 2.4.

Perspektivní projekce je proto vhodnější pro realistickou grafiku, zatímco ortografická projekce se hodí například pro architektonické aplikace, kde je rozhodující zachování velikostí objektů a úhlů mezi nimi v okamžiku projekce.

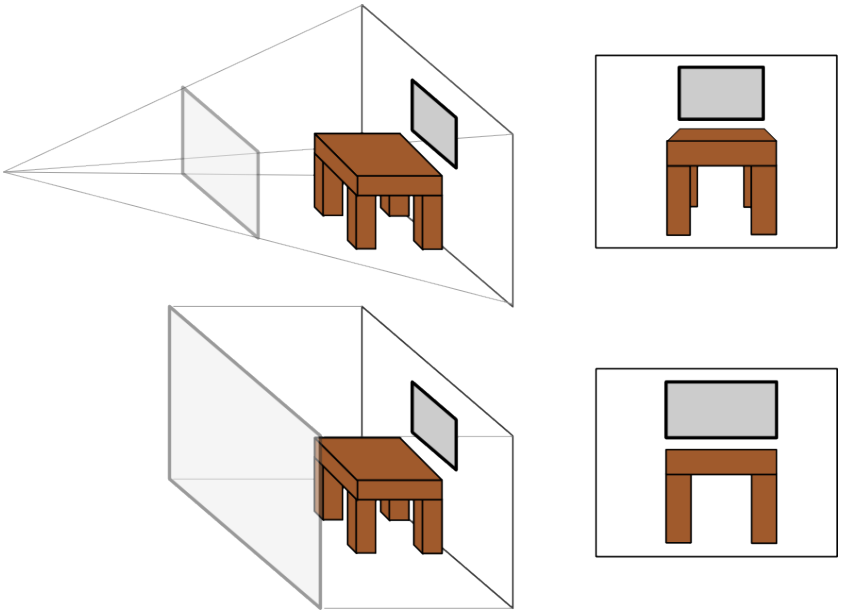
Objekty či jejich části umístěné mimo definovaný objem pohledu jsou navíc postupně ořezány podle šesti ořezových rovin. Aplikace perspektivy sestává z dělení souřadnic hodnotou w vedoucí k získání normalizovaných souřadnic zařízení.

Zobrazovací transformace (viewport transformation) neboli namapování transformovaných souřadnic objektu na souřadnice okna. Lze tak upravit dimenze zobrazení, například zvětši či zmenšit obraz[3]

Pohledová a modelovací transformace vytváří společnou pohledově modelační matici (modelview matrix). Obě tyto transformace sestávají z aplikace pohledové matice na souřadnice objektu. Výsledkem aplikace jsou souřadnice kamery.

Protože se jedná o maticové násobení, které není obecně komutativní, je důležité si uvědomit, že v pevném systému souřadnic je pořadí transformací kritické a výsledek záleží

[3]Pokud není poměr stran projekce a velikosti okna v odpovídajícím poměru, může dojít ke zkreslení



Obrázek 2.4: Perspektivní a ortografická projekce (zhora dolů), upr. [9], str. 134, 136

na pořadí aplikace jednotlivých matic. Z tohoto důvodu je nutné aplikovat transformace či příkazy transformací v obráceném pořadí, než je logické pořadí transformací.

2.6 Zásobník matic

Zásobník matic je užitečný pro tvorbu hierarchických modelů, kde jsou komplikované objekty konstruovány z jednodušších. Zásobník poskytuje mechanismus pro postupné ukládání transformačních matic do paměti a jejich vyjímání a nastavování právě vyňaté matice coby aktuální. Práce se zásobníkem bývá výkonnější než práce s jednotlivými maticemi, navíc v dnešní době je obvyklé, že je zásobník implementován hardwarově.

Zásobníky pohledově modelačních matic a projekčních matic jsou oddělené. Zásobník projekčních matic bývá obvykle pouze dvojúrovňový, neboť není třeba skládat projekční matice. Zásobník pohledově modelačních matic však může zahrnovat více matic. Každá pohledová a modelační transformace vytvoří novou matici, která násobí aktuální pohledově modelační matici a výsledek se poté stává novou aktuální maticí a reprezentuje tak složenou transformaci[4].

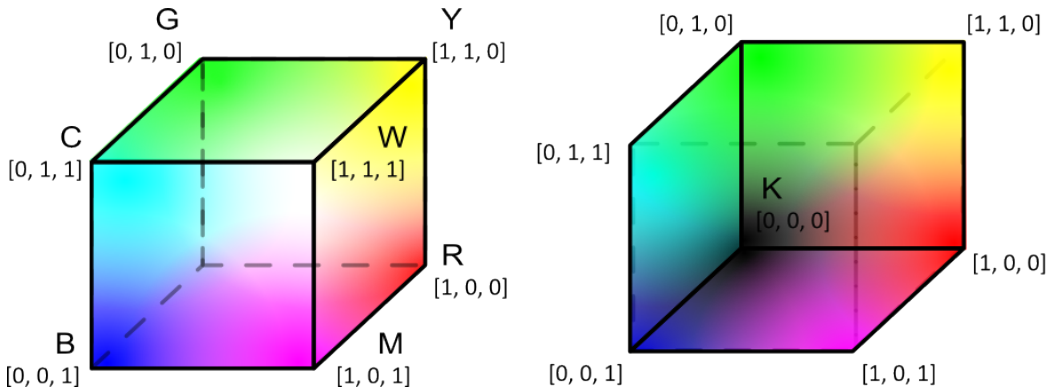
[4]Kromě pohledově modelační matice (modelview matrix) a projekční matice (projection matrix) je v OpenGL ještě třetí matice, a to texturovací, která definujíce souřadnice pro mapování textury.

2.7 Barvy

OpenGL využívá dva základní barevné módy: mód RGBA a mód indexované barva.

V módu indexované barvy se využívá barevná mapa, kde se pro každou hodnotu indexu liší hodnoty použitých barevných složek.

Mód RGBA umožňuje zadávání a uchovávání hodnoty intenzity červené (Red), zelené (Green) a modré (Blue), popřípadě hodnoty alfa neboli průhlednosti (Alpha). Oproti módu indexované barvy dává na většině systémů širší škálu barev a je flexibilnější vhledem k určitým efektům.



Obrázek 2.5: Zobrazení míchání barev v módu RBGA coby jednotkových krychlí

Rozmezí hodnot je typicky $\langle 0.0, 1.0 \rangle$, kde 0.0 je nejnižší intenzita a 1.0 nejvyšší intenzita dané barevné složky, bez ohledu na počet bitových rovin použitých pro uložení barevné složky. Hodnoty lze zadat také celočíselně, potom jsou lineárně mapovány na hodnoty čísla s plovoucí desetinnou čárkou tak, že 1.0 odpovídá maximu pro zadaný celočíselný typ a 0.0 minimu [5] [9]. Situaci znázorňuje obrázek 2.5.

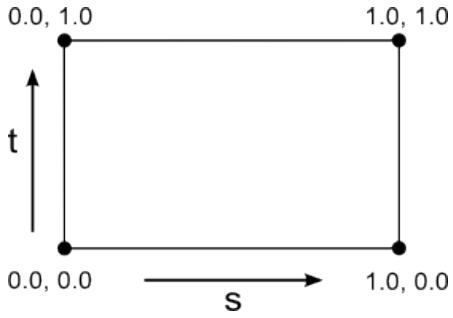
2.8 Texturování

Pojem textura označuje n-rozměrnou pixmapu. OpenGL umožňuje používání 1D, 2D nebo 3D textur. V následujícím textu se budu věnovat výhradně 2D texturám, protože právě tyto budou použity při implementaci Prohlížeče pro zobrazování fotografií.

2.8.1 Výhody texturování

Texturování se typicky používá v případech, kdy je třeba zobrazit objekt s nepříliš výrazně tvarovaným povrchem, typicky povrchy, například cihlová zeď. U zdi by bez využití texturování bylo třeba vykreslovat jednotlivé cihly, což je však náročné z paměťového, výpočetního i programátorského hlediska. S využitím texturování se vykreslování značně zjednoduší a zrychlí – omezí se na vykreslení polygonu, na který je nanesena textura zobrazující cihlovou zeď. Cenou je absence nerovností takového povrchu, ale ty lze částečně simulovat dodatečně například pomocí tzv. bump-mappingu [12].

[5] Barvu je tak možné zadat jako celé číslo bez znaménka o velikosti 1 byte, potom pro počet bitových rovin roven 8 platí, že hodnota 0 odpovídá nejnižší intenzitě barevné složky a 255 maximální intenzitě, ve finále namapované na 1.0.



Obrázek 2.6: Umístění texturovacích souřadnic; upr. orig.[2], str.151

2.8.2 Proces texturování

Při procesu texturování probíhá nanášení pixelů textury, tzv. texelů, na povrch objektu při rasterizaci podle zadaných texturovacích souřadnic. Textury je možné v OpenGL mapovat na jakékoli geometrické prvky, v Prohlížeči budou však mapovány na obdélníky odpovídajícího poměru stran.

Zdrojem pixmapy použité pro texturování může být soubor, případně lze texturu procedurálně generovat v paměti, nebo lze použít jako zdroj framebuffer. Pixmapa je mapa bodů uložených v paměti v podobě pole bez komprese, a proto je pro některé grafické formáty nutné zdrojová data po načtení následně dekomprimovat[6].

Je důležité zdůraznit, že zdroje textur bývají omezené a podporované velikosti se liší mezi jednotlivými implementacemi OpenGL. Načítání textur ze souborů probíhá čistě v režii operačního systému, nikoli v rámci knihovny OpenGL ani GLUT.

Z důvodu nedostatečné rychlosti systémové paměti a sběrnice jsou textury uloženy na GPU. Pro uložení textur použitých ve scéně slouží zvláštní část videopaměti, tj. paměti na grafickém akcelerátoru. V případně nedostatečné kapacity je možné textury přenášet postupně z operační paměti počítače. Následkem však může být značná degradace rychlosti vykreslování. Na grafickém akcelerátoru lze navíc nalézt vyrovnávací paměť pro textury, kam se ukládají často používané textury či jejich části a eliminuje se tak čtení rastrových dat z videopaměti [10].

2.8.3 Objekt textury

Pro manipulaci s texturou slouží interní struktura objekt textury. Objekt textury uchovává texturová data – pole obrazu, případné mipmapy (2.8.6), a hodnoty parametrů textury, jako je výška, šířka, šířka hranice, vnitřní formát, rozlišení složek, filtry, módy obalu, barva hranice, priorita textury. Objekt textury je nejrychlejší cestou k aplikaci textury, umožňuje práci s více texturami a vracení se k texturám již nahraným do zdrojů textur bez nutnosti opětovného načítání obrazu textury.

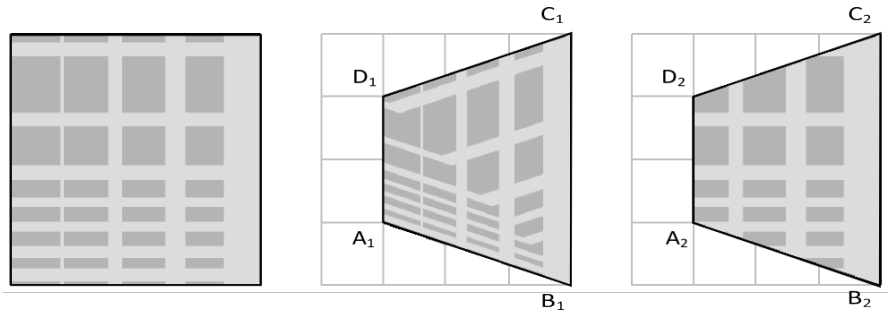
2.8.4 Přiřazování souřadnic textur

Dvourozměrné textury jsou čtvercová či obdelníková pole barev, mapovaná nejčastěji na polygony polygonálního modelu.

[6]OpenGL však umožňuje vytváření a používání komprimovaných textur, přičemž vnitřní formát komprimovaných textur se může lišit mezi implementacemi OpenGL

Pro texturovanou scénu je nezbytné zadat pro každý vrchol nejen jeho souřadnice v prostoru, ale i souřadnice textury. Souřadnice textury jednoznačně určují mapování texelu texturové mapy na tento vrchol. Texturovací souřadnice pro 2D textury mají formát (s, t), kde s,t typicky nabývají hodnot v intervalu $\langle 0; 1 \rangle$. Zadáním hodnoty v intervalu $(0; 1)$ dojde k použití pouze části textury v daném poměru a směru. Zvláštním případem jsou hodnoty texturovacích souřadnic větší než 1, v takovém případě dojde k opakování textury v zadaném směru [2].

V případě Prohlížeče se jedná o ideální situaci, kdy je pravoúhlá textura (data fotografie) mapována na pravoúhlý model (obdélníkový podklad), a to navíc v odpovídajícím formátu. Jednotlivým vrcholům obdélníku budou přiřazeny texturovací souřadnice (s, t), kde s a t nabývají hodnot z $\{0, 1\}$ dle obrázku 2.6.

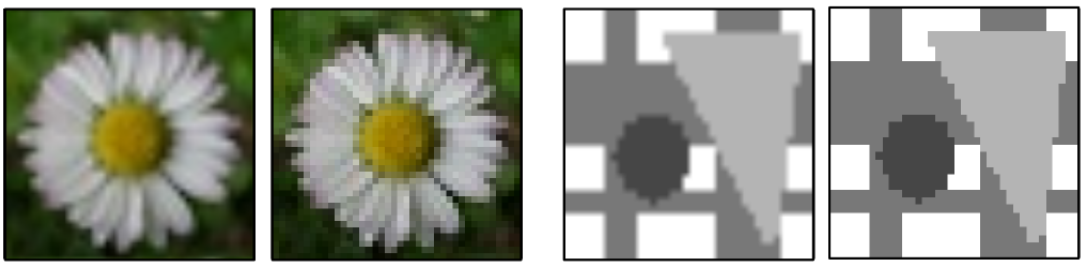


Obrázek 2.7: Použití různých texturovacích souřadnic u tvarově odlišných polygonů

Na obrázku 2.7 je znázorněno nanášení textury za pomocí různých texturovacích souřadnic. Je dobře patrné, že pro obrazec tvarově neodpovídající původní textuře, se zadáním maximálních texturovacích souřadnic $A_1 = [0.0, 0.0]$, $B_1 = [1.0, 0.0]$, $C_1 = [1.0, 1.0]$, $D_1 = [0.0, 1.0]$ se vzor textury zdeformuje. Zadáním vhodných texturovacích souřadnic $A_2 = [0.25, 0.25]$, $B_2 = [1.0, 0.0]$, $C_2 = [1.0, 1.0]$, $D_2 = [0.25, 0.75]$ ale zůstane vzor textury zachován a čistě vizuálně se jedná o tvarově odpovídající výřez z použité textury.

2.8.5 Filtrování textur

Po namapování textury na polygon a poté transformaci do souřadnic obrazovky odpovídá jeden pixel obrazovky jednomu texelu pouze v ojedinělých případech. Ve většině případů dojde k situaci, kdy pixel koresponduje s částí texelu (při zvětšení) nebo s více texely (při zmenšení). Je nutné stanovit, jaké texely použít a jak zjistit hodnotu pixelu [9].



Obrázek 2.8: Filtrování textur

OpenGL umožňuje zadat dvě možnosti filtrování:

NEAREST použije se texel, jehož souřadnice leží nejblíže středu pixelu

LINEAR vážený lineární průměr z pole 2×2 texelů ležících nejblíže středu pixelu

Tyto možnosti jsou různě výpočetně náročné a obvykle se znatelně liší rovněž kvalitou výsledného obrazu. Obecně lze říci, že určování lineárního průměru je náročnější [9]. Na obrázku 2.8 můžete pozorovat nanesení různých textur a jejich filtrování při zvětšování rozměru polygonu. Při lineárním filtrování je znatelné průměrování zejména u původně ostře ohraničených ploch, které se po filtrování jeví silně neostré. U textur tvořených fotografickým materiálem se naopak toto filtrování jeví jako velice vhodné, neboť použití metody nejbližšího texelu má za následek vznik rušivých nespojitostí.

2.8.6 Mipmapy

Mipmapami je nazývána posloupnost předdefinovaných textur se zmenšujícím se rozlišením. Podle vzdálenosti objektu od kamery, a tedy velikosti objektu v pixelech, se automaticky mění i aktuálně používaná a namapovaná textura. Mipmapy je vhodné použít v situaci, kde otexturované objekty mění svou vzdálenost od místa umístění kamery. Při pouhém filtrování (2.8.5) by mohly vznikat rušivé vizuální prvky například při pohybu objektů – míhání a blikání. Aby bylo možné mipmapy použít, je nutné poskytnout texturu ve všech velikostech daných mocninou čísla 2 od nejmenší textury 1×1 pixelů po největší texturu, která bude použita [9]. Obrázek 2.9 zachycuje mipmapy pro největší texturu o rozměrech 64×64 – postupně 32×32, 16×16, 8×8, 4×4, 2×2 a nakonec 1×1 pixelů.

Obrázek 2.9: Mipmapy

Mipmapy lze tvořit programově s použitím různých filtrů, anebo lze použít automatické generování mipmap. Při použití mipmapování lze provádět interpolaci mezi dvěma texturami (vybere se nejbližší větší a nejbližší menší textura): je vybrán nejbližší texel z nejbližší textury; bilineární interpolace nejbližších texelů nejbližší textury; lineární interpolace nejbližších texelů obou textur; interpolace pro výpočet barev texelů obou textur, výsledná barva se získá interpolací mezi dvěma předešlými, tedy se jedná o trilineární interpolace. Poslední zmíněný filtr je nejpomalejší, avšak poskytuje nejlepší výsledky [12].

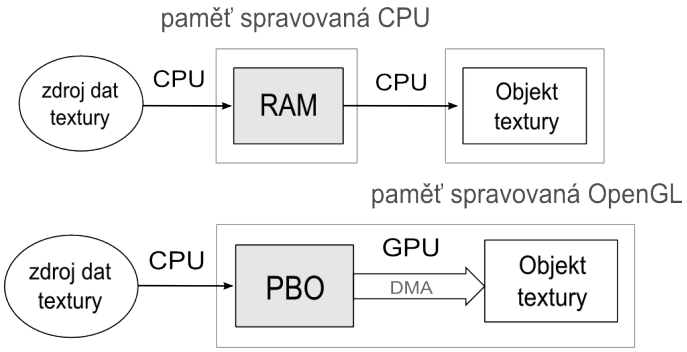
2.8.7 Nahrazování textury

Vytvoření nové textury může být výpočetně náročenější než nahrazení textury či její části novou informací. V jistých situacích je proto vhodné vytvořit jedinou texturu a postupně nahrazovat její data novými obrazy, například z videozáznamu [9].

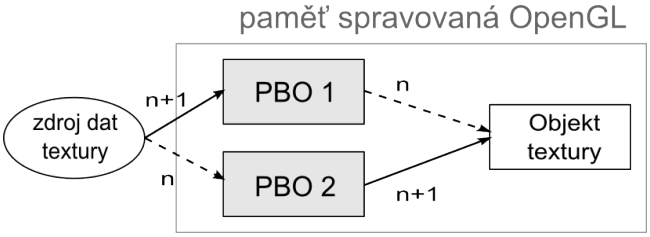
2.9 PBO

Pro efektivnější práci s daty poskytuje OpenGL jako rozšíření dvě speciální struktury, objekt paměti vrcholů (vertex buffer object, VBO) a pro uchovávání obrazových dat analogicky objekt paměti pixelů (pixel buffer object, PBO). PBO využívá aplikační programové rozhraní VBO, hlavní rozdíl je v možnosti určení přenosu pixelů. PBO se může nacházet v jednom ze dvou režim – načítání z videopaměti do RAM (například zpětné čtení framebufferu) a nahrávání z RAM do videopaměti (například nahrávání textury).

Hlavní výhodou PBO je rychlost přenosu mezi pamětí a grafickou kartou za použití přímého, asynchronního přístupu DMA do paměti. Při obvyklém načítání obrazových dat textury se vše odehrává v režii CPU. Při využití PBO se procesor účastní pouze přenosu dat do PBO. O zkopírování dat z PBO do objektu textury se dále postará grafický procesor a OpenGL [1]. Rozdíl mezi klasickým načtení obrazových dat texty a vytvoření objektu textury a použitím PBO je zřetelný z obrázku 2.10.



Obrázek 2.10: Rozdíl mezi klasickým přístupem a použitím PBO [1]



Obrázek 2.11: Použití více PBO pro načítání textur, [1]

Pro další urychlení načítání textur je možné využít více objektů PBO. Obrázek 2.11 zachycuje použití dvou objektů PBO pro maximální rychlé načítání textur. V jednom okamžiku může docházet současně k načítání obrazových dat textury ze zdrojového souboru do jednoho objektu a zároveň ke kopírování obrazových dat z druhého objektu PBO do objektu textury .

2.10 Vlákna programu

V případě Prohlížeče je použití vláken mimořádně vhodné pro účely souběžného načítání textur a vykreslování bez znatelných prodlev způsobených čekáním na dokončení načítání obrazových dat textury.

Řešením je rozdělení procesu na více vláken (multithreading). Na víceprocesorovém počítači se běh vláken může stát souběžným – vlákna běží na jednotlivých procesorech a dojde tak k značnému urychlení aplikace. Na jednoprocesorovém stroji oproti tomu dochází k simulaci paralelního běhu rychlým, dynamickým přepínáním mezi jednotlivými vlákny. Ve složitějších programech je často výhodné pracovat s více vlákny, z nichž každé může do jisté míry nezávisle na ostatních a přitom souběžně s ostatními vykonávat jistou funkcionalitu.

Na rozdíl od procesů sdílejí všechna vlákna v rámci jednoho programu systémové prostředky, kód, data a zdroje jako otevřené soubory či signály. Sdílení adresového prostoru má rovněž za příčinu transparentnost změn prováděných jednotlivými vlákny v paměti. Vlákna tak mohou operovat nad daty bez vzájemné komunikace. Každé vlákno má svůj vlastní zásobník a obsah registrů, což umožňuje vykonávat odlišný kód a udržovat lokální proměnné [5].

2.11 Synchronizace

Pro korektní práci procesu s více vlákny, kde dochází k operacím nad sdílenými zdroji, je nezbytné zajistit synchronizaci. V opačném případě by mohlo docházet k nekonzistencím zpracovávaných dat vlivem současného přístupu paralelních procesů ke sdíleným datům, případně časově závislé chybě, (race condition), typicky vznikající vlivem různé rychlosti provádění paralelních operací.

Pro vykreslovací a načítací vlákno v Prohlížeči je nezbytné řešit synchronizaci, neboť by se například mohlo stát, že by při asynchronním běhu mohlo chtít vykreslovací vlákno převzít obrazová data, která ještě nejsou kompletní, protože je načítací vlákno zatím nestačilo zcela načíst. V tomto případě načítání a předávání dat jde o typický případ synchronizačního problému nazývaného „producent-konzument“, kde producent připravuje data pro následné zpracování konzumentem. Řešením je povolit přístup ke sdíleným datům vždy pouze jednomu z nich a tímto způsobem práci s daty synchronizovat. Za účely synchronizace není vhodné použít běžné proměnné, neboť není zajištěna atomičnost přístupu a operací a k přepnutí vlákna by tam mohlo dojít například během vykonávání přiřazení – mezi jednotlivými instrukcemi procesoru. Z tohoto důvodu jsou k dispozici speciální synchronizační prostředky.

Operační systém GNU/Linux (případně kihovna SDL) poskytuje mechanismus mutex (mutual exclusion), neboli vzájemné vyloučení. Jedná se o zámek, uzamknutelný vždy pouze jedním vláknem, zatímco ostatní vlákna jsou ve svém přístupu blokována, dokud není zámek prvním vláknem opět odemknut. Dále lze využít mechanismus zvaný semafor (umožňuje specifikaci počtu současných přístupů ke sdíleným zdrojům), či podmíněnou proměnnou (v parametrech volané funkce je proměnná a příslušející zámek).

Kapitola 3

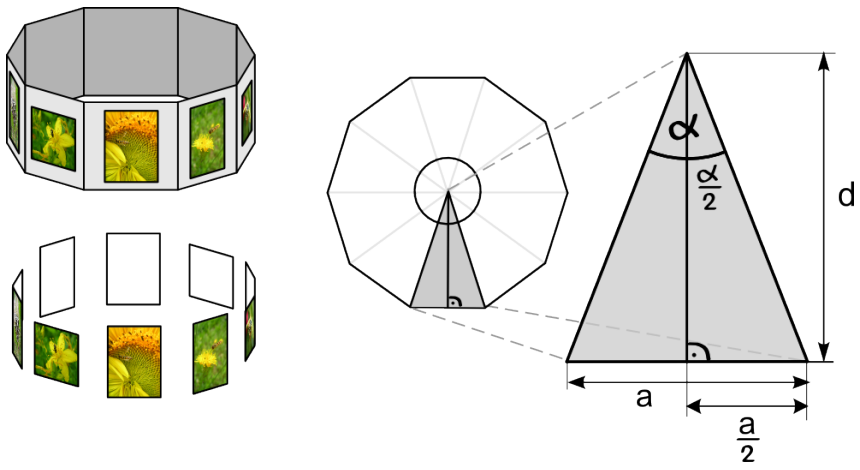
Návrh aplikace

Prohlížeč fotografií je vyvíjen v prostředí OS Microsoft Windows, konkrétně v prostředí MinGW. Záměrem je však pokusit se využít multiplatformnosti OpenGL, odladit aplikaci minimálně v prostředí OS Microsoft Windows, a v jejím rámci použít pouze multiplatformní prostředky pro budoucí rozšíření i a Linux. Program samotný je napsán v jazyce C, který pro tyto účely zcela dostačuje.

3.1 Scéna Prohlížeče

Prezentace sestává z vykreslování a transformací jednoduché scény v OpenGL. Obecně jsou fotografie samotné vizuálně zastoupeny vhodně umístěnými polygony o příslušném poměru a orientaci stran. Grafická data fotografií jsou nanesena jako textury na povrchy jednotlivých polygonů.

Fotografie jsou v prostoru umístěny na pomyslných, vertikálně umístěných stranách n-úhelníku (obrázek 3.1). Přecházení mezi jednotlivými stranami vizuálně působí jako pootáčení n-úhelníku tak, aby se aktuální stěnou, natočenou na kameru, stala následující, či předchozí stěna. Jednoduchá animace takových pootáčení prstence zajišťuje zvýšení vizuální přitažlivosti prezentace.



Obrázek 3.1: Umístění fotografií do scény

Je nutné vypočítat umístění jednotlivých stěn kolem středu, konkrétně vzdálenost strany daného rozměru od středu prstence.

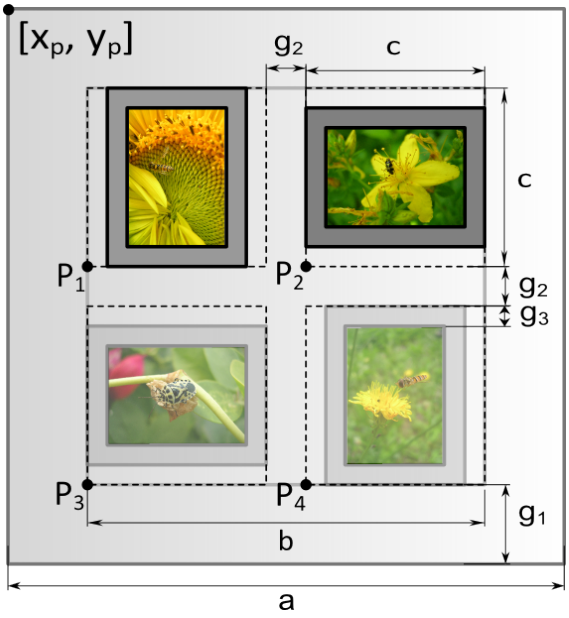
Obecný vzorec pro zjištění této délky je $d = \frac{\frac{a}{2}}{\tan(\frac{\alpha}{2})}$, kde $\alpha = \frac{360}{n}$, potom pro stupně platí:

$$d = \frac{a}{2 \cdot \tan(\frac{180}{n})} \tag{3.1}$$

a pro radiány analogicky:

$$d = \frac{a}{(2 \cdot \tan(\frac{\pi}{n})} \tag{3.2}$$

Protože je vhodné poskytnout uživateli možnost prohlížet více fotografií současně, lze přecházet mezi dvěma základními režimy procházení fotografií. Zobrazováním po jednotlivých fotografiích a po sadách více fotografií. Sadami jsou chápána pole fotografií $n \times n$, dostačující je umístění 2×2 a 3×3 fotografií na aktuální stranu. Pro elegantnější reprezentaci fotek je příhodné fotografie volitelně orámovat neutrální barvou (černá, bílá, odstíny šedé) v jednotném stylu. Zároveň je důležité zvolit nejen vhodné odsazení oblasti fotografií od okrajů stran prstence, ale i vzájemné odsazení jednotlivých fotografií při zobrazení sady fotografií na jedné straně prstence.



Obrázek 3.2: Umístění fotografií na stránce

Rozmístění fotografií 2×2 je znázorněné na obrázku 3.2. Pro souřadnice bodu P_n platí vztahy:

$$x_n = x_p + g_1 + (n-1) \cdot (c + g_2) \tag{3.3}$$

$$y_n = y_p - (g_1 + c + (n-1) \cdot (c + g_2)) \tag{3.4}$$

Nabízí se zpřístupnit uživateli možnost nastavení parametrů scény, například barvy pozadí, stran prstence, rámů fotografií, typu osvětlení scény, materiálové vlastnosti či další vizuální podoby změny scény prostřednictvím grafického uživatelského rozhraní. Vzhledem k zamýšlenému širokému spektru voleb je užitečné umožnit uložení aktuální konfigurace scény Prohlížeče do souboru a analogicky také uloženou konfiguraci načíst a aplikovat.

3.2 Grafické uživatelské rozhraní

Knihovna GLUT poskytuje skutečně minimalistický přístup k tvorbě grafického uživatelského rozhraní. Kromě možnosti otevření více oken či podoken nabízí pouze použití takzvaných vyskakovacích (pop-up) menu. Vyskakovací menu se nazývají také kontextová menu, protože často bývají vyvolána stiskem tlačítka myši nad určitým objektem v okně aplikace a může tak nabízet pouze položky menu, které mají pro daný objekt smysl. Není vhodné používat příliš mnoho menu pro různé objekty, neboť samotná existence vyskakovacího menu není uživateli nijak graficky reprezentována [11].

Naopak výhodou vyskakovacích menu je, že permanentně nezabírají mnohdy drahocenné místo na obrazovce. Pro Prohlížeč a prezentaci fotografií jsou vyskakovací menu v tomto ohledu velice žádoucí.

Prohlížeč předpokládá pouze několik typů základních objektů, se kterými lze svázat menu a příslušné volby. V základním návrhu se nabízí koncepce hlavního vyskakovací menu po kliknutí myší do libovolného prostoru scény. Celá obrazovka tak může být využita pro zobrazení scény.

Vyskakovací menu poskytnuté knihovnou GLUT je vhodné využít ve fázi vytváření aplikace, zvláště pro jednoduchost jeho vytvoření a editace. Mimo grafickou strohost je jedním z nepříjemných nedostatků vyskakovacího GLUT menu například fakt, že není možné jeho vyvolání v celoobrazovkovém režimu.

S využitím grafických prostředků OpenGL a GLUT, zde konkrétně možnosti vykreslování znaků, lze vytvořit vlastní, vizuálně přitažlivější a uživatelsky přívětivější menu a přizpůsobit je potřebám aplikace. Nabízí se vytvořit průhledné pozadí menu, které by umožňovalo současný náhled scény. Na rozdíl od GLUT vyskakovacího menu by se hlavní menu nemělo automaticky uzavřít po volbě položky. V takovém případě je totiž nutné jeho opětovné vyvolání pro uskutečnění výběru další položky. Toto je užitečné zejména při změnách například parametrů zobrazení scény. Struktura hlavního menu a vysvětlení významu koncových položek jsou zachyceny v tabulce C.1.

3.3 Načítání obrázků

Nejčastějším zdrojem obrazových dat textur v OpenGL jsou soubory formátu BMP. Data nejsou komprimována, jedná se o kompletní bitovou mapu, kdy každému pixelu je přiřazena barva pomocí RGB složek.

Nejrozšířenější formát v oblasti digitální fotografie je bezesporu JFIF, někdy označovaný rovněž jako JPEG. Fotografie reálného světa zachycují velkou škálu barev. Neobsahují ostré hrany a velké jednobarevné plochy, kde se nejvíc projevuje zkreslení vlivem komprese pomocí ztrátového algoritmu JPEG. Zástupci běžných formátů užívajících bezztrátovou metodu komprese jsou například PNG a GIF.

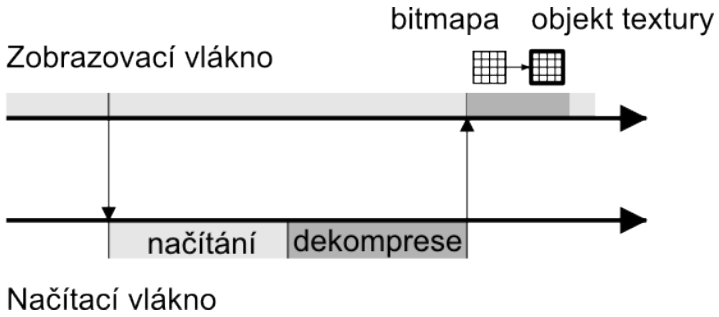
Komprimované formáty dat (JFIF, PNG, GIF) je před použitím pro účely texturování nutné dekomprimovat, často pomocí složitých algoritmů. Pro tyto účely nelze použít přímo

OpenGL, bude třeba využít některou knihovnu pro práci s obrázky, která podporuje zobrazování prostřednictvím OpenGL. V rámci Prohlížeče využíváme knihovnu DevIL, dříve známou jako OpenIL. DevIL (Developer´s Image Library) poskytuje mocné prostředky pro načítání, ukládání, konverze, manipulace, filtrování a zobrazování širokého spektra grafických formátů. Pro Prohlížeč uvažujeme jako základní podporované formáty JFIF, BMP, PNG, případně GIF.

Uživatel bude mít možnost pohodlně zadat požadovaný adresář prostřednictvím dialogového okna pro průchod a výběr zdrojového adresáře, namísto nepraktické ruční editace cesty uložené v konfiguračního souboru.

3.4 Vlákna aplikace

Bez použití vláken by se načítání textur, vykreslování a animace vykonávaly postupně. To by se znatelně projevilo na plynulosti prohlížení, prodlevy mezi snímky by byly v rámci uživatelské přívětivosti příliš velké. Pro účely Prohlížeče je nezbytné vytvořit nejméně dvě vlákna. První vlákno, „vykreslovací“, bude spolupracovat s OpenGL a starat se o vykreslování scény a animace. Druhé vlákno, „načítací“, se v případě potřeby postará o načítání obrazových dat textury z disku a bude komunikovat především s operačním systémem. Celá situace je zachycena na obrázku.



Obrázek 3.3: Použití vláken

Vykreslovací vlákno podle požadavku na zobrazení dané fotografie iniciuje načítání příslušného zdroje obrazových dat z disku načítacím vláknem. Načítací vlákno po dokončení načítání předá obrazová data či ukazatel na paměť, kde se aktuální data nacházejí. Vykreslovací vlákno poté dále manipuluje s daty, vytvoří texturovací objekt, případně texturu použije. Komunikace vláken probíhá přes proměnné, přičemž přístup k těmto proměnným musí být nutně realizován za použití některého ze zmíněných synchronizačních prostředků.

Kapitola 4

Implementace

Implementaci Prohlížeče lze rozdělit do čtyř základních oblastí: vykreslování scény, konfigurace scény, práce se souborovým systémem, práce s frontou požadavků na načítání. Tyto oblasti jsou implementovány v jednotlivých modulech, po řadě `main.c`, `konfigurace.c`, `soubory.c` a `fronta.c`.

4.1 Práce se souborovým systémem

Pro nezbytnou práci se souborovým systémem slouží funkce, proměnné a datové struktury obsažené v modulu `soubory.c` . Po spuštění programu dojde k načtení konfiguračního souboru obsahujícího cestu k adresáři s fotografiemi a naplnění datových struktur. Konkrétně je naplněna struktura `slozka`, která obsahuje položky `path`, `pole`, `obrazku`:

`path` řetězec reprezentující načtenou cestu k adresáři

`pole` pole řetězců názvů obrázkových souborů v adresáři

`obrazku` proměnná, kde je uložen počet obrázků.

4.1.1 Průchod souborovým systémem

Pro průchod souborového systému, získávání jmen souborů a podadresářů, poskytuje prostředky knihovna `<dirent.h>`. Využívají se datové struktury `DIR` reprezentující adresář a `struct dirent` reprezentující položku adresáře [4].

4.1.2 Oddělovače a řetězce

Oddělovače položek cesty se liší mezi operačními systémy. V prostředí operačních systémů Windows se používá jako oddělovač zpětné lomítko `'\'`. Zatímco na systémech typu UNIX je oddělovačem lomítko `'/'`.

Při práci s cestou je nezbytné si uvědomit, že v řetězcích jazyka C je znak zpětného lomítka `'\'` úvodním znakem pro speciální sekvence (escape sekvence), sloužící k zápisu některých zvláštních znaků. Samotné zpětné lomítko je reprezentováno zdvojeným zpětným lomítkem, tedy sekvencí `'\\'`.

V názvech souborů se může vyskytovat i mezera. Na tento fakt je třeba pamatovat při realizaci načítání cesty z příslušného konfiguračního souboru.

4.1.3 Identifikace obrázků

Pro načtení obrázků coby zdrojových souborů textur je nezbytné rozpoznat, zda se jedná o obrázek, navíc musí být podporovaného typu. Koncovka souboru se převede do jednotného formátu, co se týče velikosti písmen, využitím makra knihovny `<types.h>`. Takto upravená koncovka se poté porovná s jednotlivými koncovkami pro všechny podporovaného formáty obrázkových souborů.

4.1.4 Identifikace složek

Pro zjištění typu a stavu položek lze využít knihovnu `<sys/stat.h>`. Podstatnou je zde datová struktura `struct stat`. Tato struktura obsahuje položky st_size, st_mtime, st_mode.

Položka `st_size` udává velikost daného souboru v bajtech [4].Údaj o velikosti souboru je v programu Prohlížeče využit za účelem zjištění délky pro dynamickou alokaci řetězce pro uložení cesty.

Při procházení souborového systému je nezbytně nutné odlišit soubor od adresáře, což není možné pouze indikací nepřítomnosti souborové koncovky. Pro tento účel slouží v programu vyhodnocení položky `st_mode` struktury `stat`, která reflektuje informace o souboru a přístupová práva pomocí symbolických konstant. Adresáře je konkrétně indikován symbolickou konstantou `S_IFDIR` [4].

4.1.5 Volba zdrojové složky

Zdrojovou složku obrázkových souborů lze měnit za běhu aplikace. Uživateli je tato možnost nabídnuta v rámci hlavní nabídky. Je zobrazena nabídka pro procházení lokálním souborovým systémem. Po vyvolání koresponduje cesta s cestou načtenou z konfiguračního souboru.

Pro samotné procházení slouží dvě funkce z modulu `soubory.c`, a to `levelUp()` pro přechod na nadřazený adresář a `levelDown()` pro zanoření do vybraného adresáře. Obě funkce pouze manipulují s kopií cesty k adresáři na úrovni řetězce. Funkce `levelUp()` se projeví odebráním části řetězce odpovídající názvu posledního adresáře, který se nachází za posledním oddělovačem, zatímco funkce `levelDown()` naopak dynamicky vytvoří a uloží nový řetězec cesty vzniklý spojením řetězce původní cesty, oddělovače a názvu vybraného adresáře. Naplnění struktury `selected_dir` reflektující obsah adresáře dle nově vytvořené cesty je poté realizován funkcí `setNewDir()`.

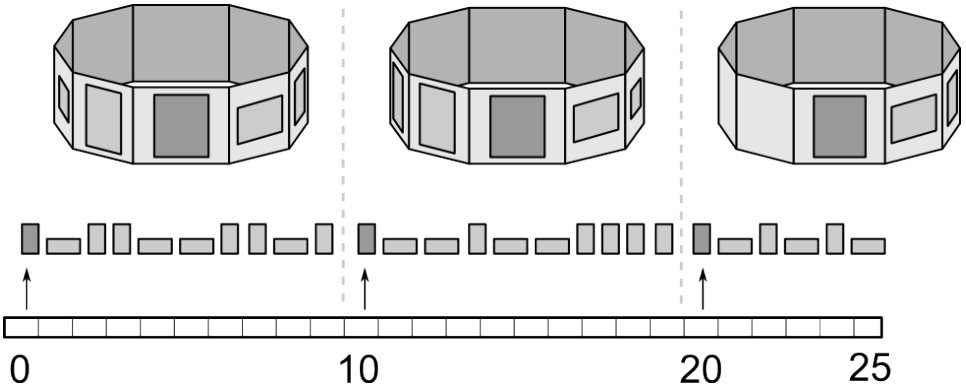
Po konečném vybrání adresáře, coby zdrojového adresáře obrázkových souborů pro zobrazení v aplikaci, je aktualizována datová struktura `slozka`.

4.1.6 Datové struktury

Načtené fotografie jsou při spuštění vloženy do dynamicky vytvořeného pole a jsou reprezentovány svými názvy. Fotografie vybrané složky jsou následně pomyslně rozděleny mezi více prstenců o pevném počtu stran zobrazujících určený počet fotografií. Situace je znázorněna na obrázku 4.1.

Při načtení fotografií ze složky proběhne výpočet počtu listů a prstenců:

```
listu = ((obrazku%(n * n)) ? (obrazku/(n * n) + 1) : (obrazku/(n * n)));
prstencu = ((listu%stran) ? (listu/stran + 1) : (listu/stran));
```



Obrázek 4.1: Rozdělení dat

kde `listu` je celkový počet listů v rámci celé aplikace, `obrazku` je celkový počet obrázků ve vybraném adresáři, `n` je uživatelem zvolený rozměr počtu fotografií na stránce, `prstencu` celkový počet prstenců, `stran` konstantní počet stran v jednom prstenci.

4.2 Grafické uživatelské rozhraní

Zdrojový soubor `main.c` implementuje samotný běh aplikace. Pracuje s knihovnami OpenGL, GLUT, DevIL, ILU a SDL. Provádí komunikaci s uživatelem, vykreslování scény, vytváření textur, práci s vlákny a událostmi.

4.2.1 Okno aplikace

Knihovna GLUT se vyznačuje jistým problémem s uzavíráním okna aplikace. Neposkytuje totiž prostředky k detekci použití zavíracího tlačítka v pravém horním rohu okna, po němž ihned následuje vyslání signálu exit. Nelze tak předem indikovat ukončení programu a provést ukončovací rutiny jako je dealokace zdrojů. V rámci aplikace jsou z tohoto důvodu uživateli nabídnuty jiné metody ukončení (stiskem klávesy Escape, zvolením položky Exit v menu).

Souvisejícím problémem je práce s více okny. Při uzavření jednoho okna a vyslání signálu exit celé aplikaci dojde k ukončení celé aplikace a uzavření všech oken. Odstranit tento nedostatek by bylo údajně možné pouze úpravou zdrojových kódu knihovny GLUT [6]. Proto je lepší více oken nepoužívat, jako se to nabízí například k vytváření dialogových oken či menu a tyto vytvářet v rámci vykreslování scény.

4.2.2 Menu

Pro vytvoření prvů grafického uživatelského rozhraní, a tedy i menu a dialogových oken, je nutné použití grafických primitiv OpenGL, vykreslování znaků pomocí knihovny GLUT a transformací za použití zásobníku modelově-pohledových matic, aby se dosáhlo vhodného umístění menu ve scéně.

4.3 Interakce s uživatelem

Aplikace je implementována tak, aby ji bylo možné pohodlně ovládat jak myší, tak klávesnicí. Knihovna GLUT poskytuje vhodné základní prostředky pro obsluhu událostí spojených s uživatelskými vstupy.

4.3.1 Obsluha událostí klávesnice

V rámci knihovny GLUT existují dva typy callback funkcí spojených s obsluhou klávesnice. Tyto dva typy se liší hodnotou vrácenou stisknout klávesou. Pomocí `glutKeyboardFunc()` lze zaregistrovat funkci pro obsluhu kláves generujících ASCII kód, jako jsou alfanumerické klávesy, ale i mezerník, Enter, BackSpace nebo Escape. Obslužné funkci je v parametru předána ASCII hodnota stisknuté klávesy.

Druhý typ funkce obsluhuje tzv. speciální klávesy, které negenerují ASCII hodnotu, ale kód stisknuté klávesy. Typicky se jedná o funkční a kurzorové klávesy. Obslužná funkce se registruje pomoc funkce `glutSpecialFunc()` [11].

V Prohlížeči jsou speciální klávesy, konkrétně kurzorové šipky využity pro iniciování rotace prstence ve scéně zadaným směrem podle směru stisknuté šipky a orientace prstence. Přepínání mezi jednotlivými prstenci je rovněž možné pomocí kurzorových šipek.

4.3.2 Obsluha událostí myši

Mezi události, které je možné pomocí zaregistrovaných callback funkcí obsluhovat patří kliknutí tlačítka myši a pohyb myši. Callback funkci pro obsluhu události spojené s kliknutím tlačítka myši jsou předány jako parametry informace o pozici myši, konkrétním tlačítku (levé, prostřední, pravé) a stavu tlačítka (puštěné, stisknuté).

Pomocí knihovny GLUT lze zachytit také pohyb myši, a dvou druhů: aktivní a pasivní. Aktivním pohybem je chápán pohyb myši během trvání stisku tlačítka. Pasivní pohyb myši analogicky znamená pobyh myši bez současně stisknutého tlačítka. Pro registraci callback funkcí pro obsluhu těchto událostí slouží funkce `glutMotionFunc()` pro aktivní pohyb myši a `glutPassiveMotionFunc()` pro pohyb pasivní.

Obsluha aktivního pohybu myši je v aplikaci využita pro určování rotace prstence ve scéně, zatímco pasivní pohyb myši je třeba zachycovat v případě zobrazení menu a označení aktivní položky, neboť zde se kurzor myši pohybuje mezi jednotlivými položkami bez stisknutého tlačítka [3].

4.4 Vykreslování scény

4.4.1 Konfigurace scény

Soubor modulu `konfigurace.c` obsahuje proměnné použité pro uchovávání vizuální podoby aplikace a funkce pro jejich načtení či uložení.

Uživateli je nabídnuta možnost měnit vzhled aplikace, konkrétně barvu papíru stránek v prstenci, barvu pozadí, barvu rámečků, styl zobrazení stránek, počet fotografií na stránku, či orientaci prstence. Zvolenou konfiguraci je možné uložit. Načte se automaticky při dalším spuštění aplikace.

Po spuštění programu je funkcí `loadConfig()` načten obsah proměnných z konfiguračního souboru, kde je zaznamenána poslední uložená vizuální podoba scény. V případě neúspěchu při otevření souboru jsou proměnné inicializovány výchozími hodnotami.

Uživatel může obsah proměnných měnit za běhu aplikace prostřednictvím hlavní nabídky, případně aktuální vizuální konfiguraci uložit. Toto uložení je realizováno funkcí `saveConfig()`.

4.4.2 Vykreslování prstence

Pro vykreslování horizontálně umísteného prstence je klíčovým využití postupné rotace kolem osy `y` o úhel $angle = \frac{360}{n}$, kde `n` je počet stran, a postupné vykreslování jednotlivých stran sestávajících z listů nebo samostatných fotografií. Pro vertikálně umístěný prstenec je situace obdobná, rotace však probíhá kolem osy x.

4.4.3 Načtení obrazových dat

Obrázky jako zdroje textur jsou načítánya automaticky dekomprimovány do bitmapové podoby prostřednictvím funkcí knihovny `<IL/IL.h>`. Protože může nastat problém se stranovou orientací obrázků, tj. některé soubory se mohou načíst zrcadlově, je nutné použít funkce `ilEnable()` a `ilSetInteger()` [14].

```
ilEnable(IL_ORIGIN_SET);
ilSetInteger(IL_ORIGIN_MODE, IL_ORIGIN_UPPER_LEFT);
```

Nezbytná je následná změna rozměrů načteného zdrojového obrázku. Změnu rozměrů lze provést prostřednictvím speciální funkce OpenGL, zde je však nezbytné operovat s pamětí, přesněji zadat nové umístění textury. Jednodušší variantou je využití funkce knihovny `<IL/ILU.h>` a funkcí `iluImageParameter()` a `iluScale()`:

```
iluImageParameter(ILU_FILTER, ILU_LINEAR);
iluScale(dx, dy, ilGetInteger(IL_IMAGE_DEPTH));
```

kde `iluImageParameter()` udává typ filtrování pro změnu velikosti, zde lineární a `iluScale()` změní obrázek o původních rozměrech x a y na obrázek o nových rozměrech dx, dy. Hloubka zůstane stejná díky zadání hodnoty hloubky původního obrázku. Zároveň dojde k uložení hodnot výšky, šířky a zejména výpočtu poměru výšky a šířky, který je nezbytný pro vykreslení podkladového polygonu pro danou fotografii [14].

4.4.4 Animace

GLUT poskytuje možnost registrace callback funkce[1] časovače (Timer). Registrovaná funkce je spuštěna vždy po uplynutí zadaného počtu milisekund a nabízí se pro účely programování animací.

Dále lze využít callback funkci volanou v případě nečinnosti aplikace (Idle). Funkce Idle se poté provede v momentě, kdy dojde k přidělení volného času procesoru dané aplikaci, avšak není registrována žádná událost [11].

Animace je realizována pomocí časovače knihovny GLUT, konkrétně funkcí `glutTimerFunc()`, kde parametry je časový interval před voláním funkce časovače na kterou ukazuje druhý

[1] callback funkce jsou funkce, volané po výskytu příslušné události

parametr a třetím parametrem je libovolná hodnota která může sloužit například k rozhodnutí podmínky ve funkci. Při animaci scény specifikuje hodnota value směr pohybu a typ animace (podle číselné klávesnice 2 pro pohyb dolů, 8 nahoru, 6 vpravo a 4 vlevo).

Animace může být vyvolána jak stiskem klávesy, tak pomocí myši. V případě posunu klávesami se během animace ignorují násobné stisky kláves. Výjimkou však jsou klávesy příslušející směru animace, jejichž stisknutí během animace má za následek zkrácení časového intervalu pro funkci časovače a urychlení animace. Posun myší doprava a doleva je realizován jako uložení pozice kurzoru při stisknutí tlačítka myši a zjišťování směru následného pohybu odečítáním aktuální pozice od pozice uložené při stisku. Za předpokladu, že se myš se stisknutým tlačítkem pohybuje například vpravo od počáteční pozice, realizuje se animované jedno či více pootočení prstence směrem vpravo.

4.5 Načítání fotografií a vlákna

4.5.1 Knihovna SDL

Vzhledem k faktu, že OpenGL i GLUT zaručují přenositelnost výsledné aplikace, nabízí se zachovat tuto přenositelnost a využít při implementaci Prohlížeče multiplatformní knihovnu, poskytující funkce pro vytváření, řízení a synchronizaci vláken. Takovou knihovnou je právě multimediální knihovna SDL. SDL byla navržena pro tvorbu zejména her pro různé operační systémy. Poskytuje programátorovi široké možnosti využití – pro práci s médii (video, audio), uživatelskými vstupy, zařízeními (klávesnice, joystick, CD-ROM), časem a již zmiňovanými vlákny. Více informací a potřebnou dokumentaci lze nalézt na stránkách knihovny SDL [15].

4.5.2 Prostředky synchronizace

Multimediální knihovna SDL poskytuje synchronizační mechanismy – mutex, semafor a podmíněnou proměnnou. Pro synchronizaci v případě producent-konzument se nabízí využít mutex, nebo semafor.

Mutex je jednoduchý prostředek vhodný k zamezení současného přístupu více vláken k jedné sdílené proměnné [13]. Knihovna SDL poskytuje pouze blokující funkci pro přístup k mutexu. Znamená to, že funkce pro uzamykání mutexu se neustále vykonává, dokud není možné mutex uzamknout [15].

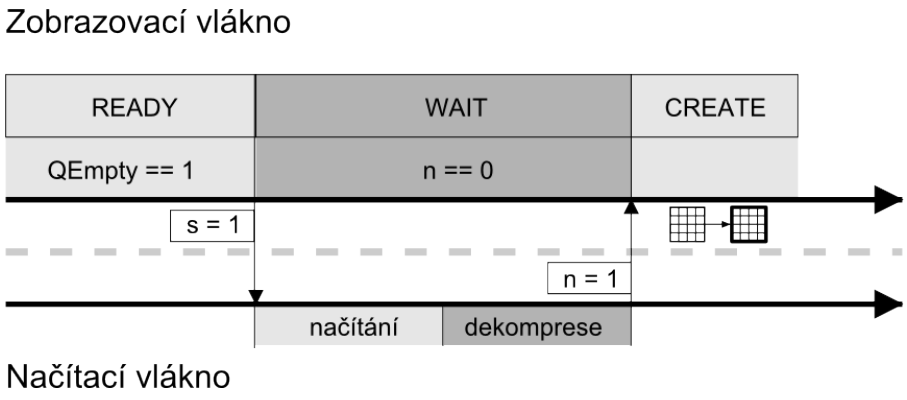
Semafor slouží k omezení počtu vláken v určené části kódu, například za účelem omezení kvůli výkonu nebo zamezení současného přístupu do tzv. kritické sekce, kde může dojít k současné manipulaci se sdílenými proměnnými. Knihovna SDL umožňuje tři funkce pro uzamknutí semaforu – `SDL_SemWait()`, `SDL_SemTryWait()`, `SDL_ SemWaitTimeout()`.

Funkce `SDL_SemWait()` pozastaví vlákno, dokud není možné semafor uzamknout, vyčkává, až hodnota semaforu se dostane do kladných hodnot. Tato funkce je blokující. Neblokující funkce `SDL_SemTryWait()` nečeká na vpuštění, ale vrátí konstantu `SDL_MUTEX_TIMEDOUT`. Varianta `SDL_SemWaitTimeout()` čeká na vpuštění po stanovený počet milisekund a poté vrací konstantu `SDL_MUTEX_TIMEDOUT`. Tato funkce však může být na některých platformách neefektivní, je-li implementována cyklem, který každou milisekundu textuje hodnotu zadaného semaforu [13].

Pro synchronizaci načítacího a vykreslovacího vlákna byl použit mechanismus mutex. Funkce pro uzamknutí je proto blokující. Kritická sekce však vždy obsahuje pouze jediný příkaz přiřazení za účelem zjištění či nastavení aktuální hodnoty, proto lze toto blokování tolerovat.

4.5.3 Komunikace vláken

Komunikace za účelem načítání probíhá mezi funkcí načítacího vlákna a glut funkcí typu Idle. Idle funkce představuje jednoduchý stavový automat a může se nacházet ve třech stavech: `READY`, `WAIT`, `CREATE`. Komunikaci mezi vlákny a stavy automatu zachycuje obrázek 4.2.



Obrázek 4.2: Komunikace vláken

`READY` provede se test, zda je ve frontě vložen požadavek na načtení fotografie, pokud je fronta neprázdná, vyjme první prvek a inicializuje načítání vláknem.

`WAIT` testuje stav načtení zadaných obrazových dat vláknem. Po dokončení načtení přejde automat do stavu `CREATE`.

`CREATE` vytvoří z nových obrazových dat objekt textury, automat přejde do stavu `READY`.

4.5.4 Fronta požadavků na načtení

Modul `fronta.c` obsahuje datovou strukturu fronty a funkce pro manipulaci s frontou: inicializaci, vložení nového prvku na konec fronty, čtení prvního prvku, odstranění prvního prvku, vyprázdnění fronty.

4.5.5 Práce s frontou

Prvky fronty jsou, za účelem zjišťování aktuálnosti požadavku při vyjmutí (vysvětleno v 4.5.6), datové struktury obsahující index do pole názvů zdrojových obrázkových souborů a hodnota čítače, která byla aktuální v momentě vkládání do fronty.

S frontou požadavků na načtení obrazových dat pracuje pouze zobrazovací vlákno. Po začátku aplikace nebo přesunu mezi prstenci se do fronty vloží indexy do pole názvů zdro-

jových souborů obrazových dat, příslušejících fotografiím na viditelných stranách prstence a vhodného počtu stránek, které je vhodné začít načítat[2].
Při pootočení prstence dojde ke dvěma akcím vzhledem k objektům textur:

Načítání všech fotografií na nově viditelné strany, přesněji vložení jejich indexů z pole názvů zdrojových souborů do fronty požadavků na načtení.

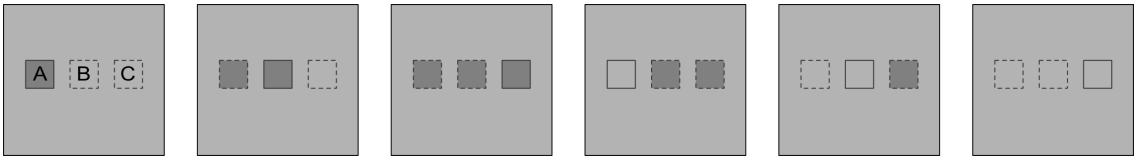
Smazání objektů textur obsahujících obrazová data, která již není třeba zobrazovat. K tomuto účelu disponuje OpenGL funkcí `glDeleteTextures()`, která smaže n objektů textur, jejichž jmény jsou prvky zadaného pole. Při pokusu o smazání neexistujících textur nebo textur se jménem rovným nule jsou ignorovány a nezpůsobí chybu [9]. Tento fakt znamená, že není nutné testovat textury před požadavkem na smazání.

4.5.6 Problém aktuálnosti požadavku na načtení

Může se vyskytnout situace, kdy je požadavek na načtení přítomen ve frontě a přitom již byl prstenec pootočen tak, že už není třeba tuto fotografii zobrazit. Textura je tak uchovávána zbytečně. Toto může nastat vlivem dvou faktorů: rychlosti otáčení prstence a rychlosti načítání jednotlivých fotografií, typicky jsou-li zdrojové obrázky značné velikosti. Této situaci předejdeme testováním aktuálnosti požadavku na načtení obrazových dat dané textury vzhledem k poslednímu požadavku na smazání této textury. Kontrolním prvkem je pole, kde se uchovává hodnota čítače zaznamenaná v okamžiku posledního smazání textury. Požadavek na načítání je kontrolován hned dvakrát: při vyjmutí z fronty, tedy před načítáním, a před zpracováním obrazových dat na texturu. Pokud je hodnota čítače v požadavku na načtení nižší, než hodnota v poli, je požadavek vyhodnocen jako neaktuální a není zpracován.

4.5.7 Vizuální znázornění nenačtených textur

Vizuální znázornění načítání fotografie je provedeno pomocí nahrazování textury (vysvětleno v 2.8.7). Na počátku je vytvořena jednobarevná podkladová textura. Poté ve funkci časovače periodicky v daných intervalech dochází k překreslování obrazových dat této textury, konkrétně v dané implementaci se postupně vykreslí tři tmavší čtverce na třech určených pozicích v rámci základní světlé textury a poté se opět postupně překreslí třemi čtverci v barvě základní textury, jak je znázorněno na obrázku 4.3. Pro tuto texturu se neprovádí výpočetně náročnější lineární filtrování, neboť ve finále obsahuje tato textura jednoduché geometrické obrazce, kde je zachování ostrosti hran naopak velice vhodné.



Obrázek 4.3: Animace nahrazováním textury

[2]Pro režim zobrazení 1 fotografie na stránku se uchovává maximálně 10 textur, což není nijak závratný počet a z hlediska uživatelské příjemnosti je lepší načíst všechny obrázky pro aktuální prstenec a vytvořit a používat příslušné texturové objekty bez jejich znovuvytváření či mazání.

Kapitola 5

Závěr

5.1 Zhodnocení výsledků

Cílem této práce bylo vytvořit uživatelsky přívětivou a jednoduchou aplikaci pro zobrazování fotografií, využívající multiplatformní knihovnu OpenGL a výhody 3D akcelerace. Tohoto cíle bylo úspěšně dosaženo, používání výsledné aplikace je dostatečně intuitivní a prezentace fotografií ve 3D scéně značně vizuálně atraktivní.

Za účelem zajištění plynulosti zobrazování scény bylo implementováno načítání obrázkových souborů pomocí načítacího vlákna. V rámci řešení problému producent-konzument byla rovněž provedena nutná synchronizace obou vláken aplikace prostřednictvím synchronizačních prostředků, poskytovaných knihovnou SDL.

Pro zvýšení uživatelské přívětivosti byly implementovány vlastní prvky grafického uživatelského rozhraní, konkrétně hlavní menu aplikace pro volbu vizuálního nastavení zobrazování a dialogové okno pro procházení souborového systému a výběr zdrojového adresáře.

Výsledná aplikace pracuje s fotografiemi, respektive obrázkovými soubory, v libovolném rozlišení a podporovaných formátech JFIF, PNG, GIF a BMP.

Aplikace byla vyvíjena na operačním systému Microsoft Windows Vista, kde byla rovněž průběžně testována pro různé zdrojové adresáře fotografií, zejména pak na fotografiích o značných rozměrech stran, konkrétně 4752×3168 px. Dále byla aplikace testována obrázky pro jednotlivé podporované formáty a možné podoby formátových koncovek, týkající se velikosti písmen. Aplikace byla rovněž testována v prostředí Microsoft Windows XP. Protože je aplikace psána s využitím multiplatformních knihoven a funkcí, je připravena pro budoucí odladění a testování na dalších platformách, například Linux.

Při testování na starších počítačích bylo pozorováno výrazné zpomalení běhu aplikace, projevující se zejména v rychlosti animací. Toto je zřejmě způsobeno skutečností, že načítání, a zejména dekomprimace (nejvýrazněji u formátu JFIF), je velice náročná na výkon CPU.

5.2 Možnosti rozšíření

Pro budoucí rozšíření je možné uvažovat implementaci možnosti otáčení fotografií v rámci zobrazování v Prohlížeči. V dosavadním řešení Prohlížeče se předpokládá, že fotografie jsou uživatelem předem otočeny.

Užitečnou by byla také možnost uchovávání a volby z více cest k různým adresářům s fotografiemi, tedy udržování virtuálních alb.

Prvkem zvyšujícím uživatelskou přívětivost by dále mohla být lišta rychlé navigace ve vybrané složce fotografií, zobrazující ikony obsažených fotografií, kde by byla rovněž vyznačena aktuální pozice prohlížení a uživateli by byla poskytnuta možnost pohodlného vybrání nové pozice.

Literatura

[1] Ahn, S. H.: OpenGL Pixel Buffer Object (PBO). [online; cit. 28. ledna 2010], 2007.
URL http://www.songho.ca/opengl/gl_pbo.html

[2] Astle, D.; Hawkins, K.: *Beginning OpenGL Game Programming (Game Development Series)*. Course Technology PTR, 2004, iSBN 1-59200-369-9.

[3] Fernandes, A. R.: GLUT tutorial. [online; cit. 24. dubna 2010].
URL http://www.lighthouse3d.com/opengl/glut

[4] Herout, P.: *Učebnice jazyka C – 2. díl*, ročník 3. České Budejovice: KOPP, 2007, iSBN 978-80-247-2254-2.

[5] Mitchell, M.; Oldham, J.; Samuel, A.: Advanced Linux Programming. [online; cit. 28. ledna 2010], 2001.
URL http://www.advancedlinuxprogramming.com

[6] Ploskirev, A.: OpenGL FAQ / 3 GLUT. [online; cit. 24. dubna 2010], 2000.
URL http://www.opengl.org.ru:8000/docs/faq/glut.htm

[7] Pokorný, P.: *DirectX začínáme programovat*, ročník 1. Praha: Grada Publishing a.s., 2008, iSBN 978-80-247-2254-2.

[8] Poliščuk, R.: Přednáška Počítače a grafika. [online; cit. 10. března 2010], 2008.
URL http://autnt.fme.vutbr.cz/poliscuk/VPG/pg10.pdf

[9] Shreiner, D.; Woo, M.; Neider, J.; aj.: *OpenGL Průvodce programátora*, ročník 1. Praha: Computer Press a.s., 2006, iSBN 80-251-1275-6, anglická elektronická verze dostupná na URL <`http://www.glprogramming.com/red`>.

[10] Tišnovský, P.: Grafické karty a grafické akcelerátory (20). [online; cit. 21. ledna 2010], 2003.
URL
http://www.root.cz/clanky/graficke-karty-a-graficke-akceleratory-20/

[11] Tišnovský, P.: Seriál Tvorba přenositelných grafických aplikací využívajících knihovnu GLUT. [online; cit. 8. dubna 2010], 2003.
URL http://www.root.cz/serialy/
tvorba-prenositelnych-grafickych-aplikaci-vyuzivajicich-knihovnu-glut

[12] Tišnovský, P.: Grafická knihovna OpenGL. [online; cit. 1. února 2010], 2004.
URL http://www.root.cz/serialy/graficka-knihovna-opengl/

[13] Turek, M.: SDL: Hry nejen pro Linux (22). [online; cit. 10. března 2010], 2005.
URL http://www.root.cz/clanky/sdl-hry-nejen-pro-linux-22

[14] WWW stránky: DevIL – A full featured cross-platform Image Library. [online; cit. 28. ledna 2010].
URL http://openil.sourceforge.net

[15] WWW stránky: Simple DirectMedia Layer. [online; cit. 10. března 2010].
URL http://www.libsdl.org

[16] WWW stránky: Wikipedie - článek o GPU. [online; cit. 8. dubna 2010].
URL http://cs.wikipedia.org/wiki/GPU

[17] WWW stránky: FotoViewr. [online; cit. 28. ledna 2010], c2008.
URL http://www.fotoviewr.com

[18] WWW stránky: Cooliris. [online; cit. 28. ledna 2010], c2010.
URL http://www.cooliris.com

[19] WWW stránky: Epicreal Software. [online; cit. 28. ledna 2010], c2010.
URL http://www.epicreal.com

[20] WWW stránky: Visions. [online; cit. 28. ledna 2010], c2010.
URL http://www.twinsvisions.com

Seznam zkratek

API	Aplikační rozhraní
BMP	Microsoft Windows Bitmap
COM	Component Object Model
CPU	Central Processing Unit
DevIL	Developer's Image Library
DMA	Direct Memory Access
GIF	Graphic Interchange Format
GLUT	OpenGL Utility Toolkit
GPU	Graphic Process Unit
JFIF	JPEG File Interchange Format
JPEG	Joint Photographic Expert Group
OpenGL	Open Graphics Library
OS	Operation System (Operační systém)
PBO	Pixel Buffer Object
PNG	Portable Network Graphics
RGB	Red Green Blue
RGBA	Red Green Blue Alpha
SGI	Silicon Graphics Inc.
SDL	Simple Media Layer
VBO	Vertex Buffer Object

Seznam příloh

Dodatek A

Obsah CD

`./doc`	programová dokumentace a uživatelský manuál
`./bin`	spustitelná aplikace a vzorové fotografie
`./src`	zdrojové kódy, Makefile, Readme
`./latex`	zdrojová data technické zprávy
`./zprava.pdf`	technická zpráva

Dodatek B

Manuál

B.1 Prohlížení fotografií

Pootočení prstence doleva, přechod na následující stranu

K: klávesa ←

M: kliknutí myši a pohyb se stisknutým tlačítkem nalevo od tohoto místa

Pootočení prstence doprava, přechod na předešlou stranu

K: klávesa →

M: kliknutí myši a pohyb se stisknutým tlačítkem napravo od tohoto místa

Změna prstence na předešlý

K: klávesa ↑

M: kliknutí myši poblíž horního okraje okna

Změna prstence na následující

K: klávesa ↓

M: kliknutí myši poblíž dolního okraje okna

Pozn.: Platí pro horizontálně orientovaný prstenec, pro vertikální jsou klávesy, kliknutí a pohyby myší analogické vzhledem k orientaci prstence: otáčení prstence je vyvoláno klávesami ↓, ↑ a pohyby myši nahoře a dole od stisknutí tlačítka, přechod mezi prstenci klávesami →, ← nebo kliknutími poblíž pravého či levého okraje okna.

Přiblížení fotografie

K: klávesa 1-9 odpovídající pořadí fotografie na stránce

Zrušení přiblížení fotografie

K: klávesa 0

Přiblížení, či oddálení scény

K: klávesa +, -

B.2 Hlavní menu

Vyvolání hlavního menu

M: stisk pravého tlačítka myši v režimu prohlížení

Otevření podnabídky položky

K: klávesa →

M: stisk levého tlačítka na nekoncové položce

Vybrání položky menu

K: klávesa Enter

M: stisk levého tlačítka na koncové položce

Nadřazená úroveň menu

K: klávesa ←

M: stisk pravého tlačítka

Opuštění menu

K: klávesa Escape

M: stisk pravého tlačítka v prostoru vně okna menu

B.3 Výběr zdrojové složky

Přechod na označený adresář

K: klávesa →

M: stisk levého tlačítka

Přechod na rodičovský adresář

K: klávesa ←

M: stisk pravého tlačítka

Změna disku

K: klávesa odpovídajícího písmena

Vybrání nového zdrojového adresáře

K: klávesa Enter

M: kliknutí myši na název aktuálně otevřeného adresáře

Dodatek C

Struktura hlavního menu

Colors	**Page**	None White Gray Red Green Blue Yellow Brown Azure Black *- volba barvy stránky*
	Frame	None White Black Gray *- volba barvy rámečku fotografií*
	Background	White Black Gray *- volba barvy pozadí scény*
Style	**Pictures**	1×1 2×2 3×3 *- počet zobrazených fotografií na stránce*
	Ring	Horizontal Vertical *- orientace prstence*
Configuration	Save scene *- uložení aktuální konfigurace scény*	
	Change directory *- vyvolání dialogu pro výběr zdrojové složky*	

FullScreen
- celoobrazovkový režim

Help
- zobrazení nápovědy

Exit
- konec aplikace

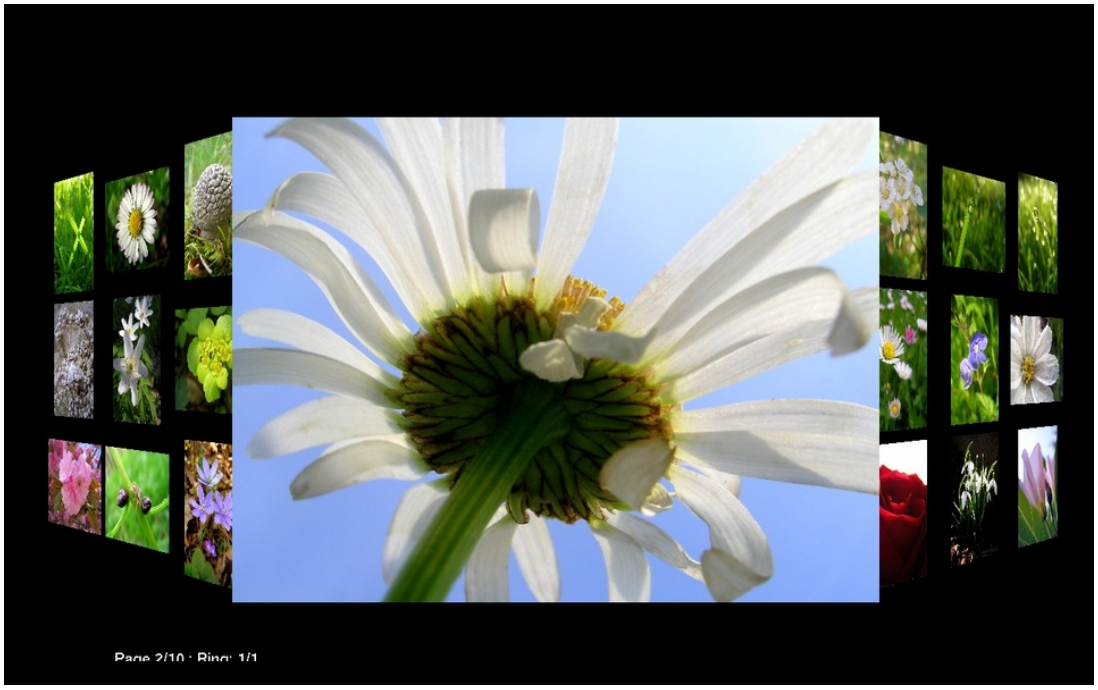
Tabulka C.1: Struktura hlavního menu

Dodatek D

Ukázky aplikace

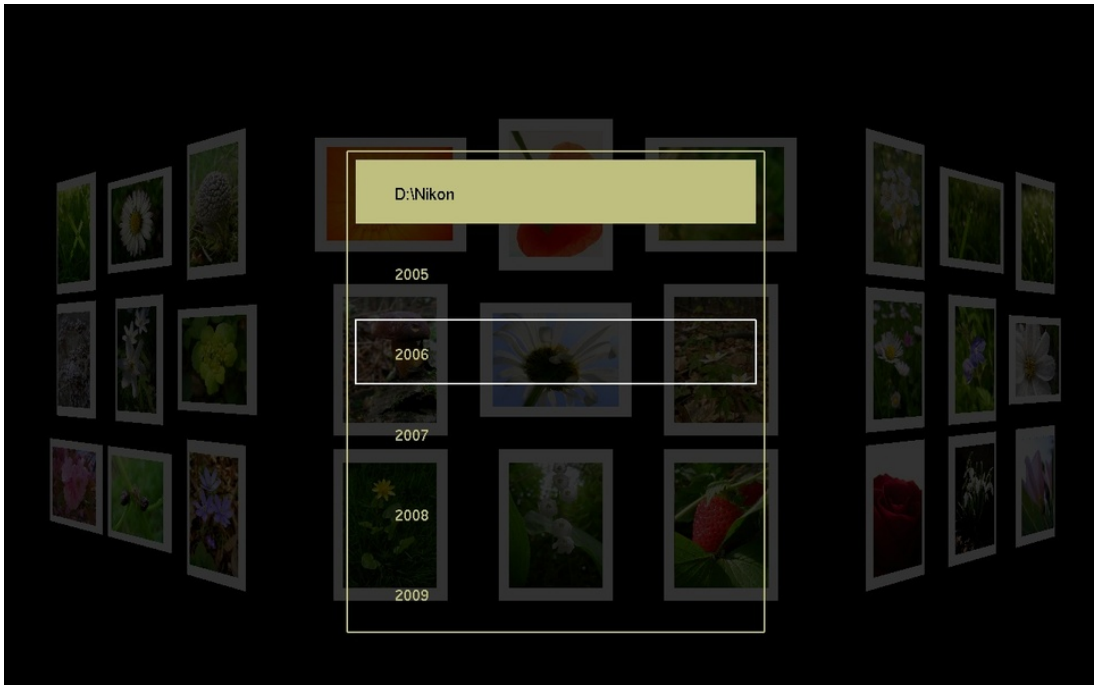
Obrázek D.1: Prstenec fotografií bez zapnutých stránek a rámečků

Obrázek D.2: Prstenec fotografií se zapnutými stránkami a rámečky



Obrázek D.3: Přiblížení fotografie

Obrázek D.4: Vertikální vykreslení prstence a hlavní menu



Obrázek D.5: Zobrazení více fotografií na stránce a dialog pro průchod adresáři